

# On Automating the Extraction of Programs from Termination Proofs\*

Fairouz Kamareddine<sup>†</sup>    François Monin<sup>‡</sup>    Mauricio Ayala-Rincón<sup>§</sup>

## Abstract

We investigate an automated program synthesis system that is based on the paradigm of programming by proofs. To automatically extract a  $\lambda$ -term that computes a recursive function given by a set of equations the system must find a formal proof of the totality of the given function. Because of the particular logical framework, usually such approaches make it difficult to use termination techniques such as those in rewriting theory. We overcome this difficulty for the automated system that we consider by exploiting product types. As a consequence, this would enable the incorporation of termination techniques used in other areas while still extracting programs.

**Keywords:** *Program extraction, product types, termination, ProPre system.*

## 1 Introduction

The Curry-Howard isomorphism [3] that establishes a correspondence between programs and proofs of specifications plays a major role in many type systems. Programming methods using the *proof as program* paradigm ensure some correctness of programs extracted from a proof of function totality and provide a logical framework for which the behaviour of programs can be analysed. Of these systems which exploit the proof as program paradigm, we mention *Second Order Functional Arithmetic (AF2)*, cf. [7, 9]) and a faithful extension of *AF2* called *Recursive Type Theory (TTR)*, cf. [16]). Both systems use equations as algorithmic specifications. In *AF2* and *TTR*, the compilation phase corresponds to formal termination proofs of the specifications of functions from which  $\lambda$ -terms that compute the functions are extracted.

Using the logical framework of *TTR*, an automated system called *ProPre*, has been developed by P. Manoury and M. Simonot [12, 11]. The automated termination problem turns out to be a major issue in the development of the system. Alongside the system, where data types and specifications of functions are introduced by the user in an ML-style, an algorithm has been designed using strategies to search for formal termination proofs for each specification. When the system succeeds in developing a formal termination proof for a specification, a  $\lambda$ -term that computes the function is given.

As mentioned in [12], the automated termination proofs in this system differ from the usual techniques of rewriting systems because they have to follow several requirements. They

---

\*This paper is an extended version of [6].

<sup>†</sup>Corresponding author. School of Mathematical and Computer Sciences, Heriot-Watt University, Edinburgh, Scotland. [fairouz@macs.hw.ac.uk](mailto:fairouz@macs.hw.ac.uk)

<sup>‡</sup>Département d'informatique, Université de Bretagne Occidentale, CS 29837, 29238 Brest Cedex 3, France. [monin@univ-brest.fr](mailto:monin@univ-brest.fr)

<sup>§</sup>Departamento de Matemática, Universidade de Brasília, Brasília D.F., Brasil. [ayala@mat.unb.br](mailto:ayala@mat.unb.br)